INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Centralité du second ordre : Calcul distribué de l'importance de noeuds dans un réseau complexe

Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Sericola, Gilles Trédan

N° 6809

Janvier 2008

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6809--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Centralité du second ordre : Calcul distribué de l'importance de noeuds dans un réseau complexe

Anne-Marie Kermarrec, Erwan Le Merrer, Bruno Sericola, Gilles Trédan

Thème NUM — Systèmes numériques
Équipes-Projets Asap et Dionysos

Rapport de recherche n° 6809 — Janvier 2008 — 24 pages

Résumé : Un réseau complexe peut être modélisé sous la forme d'un graphe représentant la relation "qui connait qui". Dans le contexte de la théorie des graphes pour les réseaux sociaux, la notion de centralité a été introduite pour mesurer l'importance relative de noeuds par rapport à une topologie donnée.

Prenons par exemple un réseau composé de grands clusters denses qui ne sont reliés que par quelques liens. Les noeuds impliqués dans ces liens sont vitaux pour conserver la connexité du graphe. Un tel phénomène peut aussi avoir des impacts sur les applications exploitant ce graphe. Connatre l'importance de tels noeuds peut s'avérer utile pour la maintenance de la topologie d'un graphe, ou pour anticiper les congestions et les déconnexions.

Beaucoup de formes de centralités ont déjà été définies. Malheureusement elles sont destinées aux graphes abstraits. Par conséquent elles sont, dans le contexte d'un système distribué, soit d'un intérêt limité (centralité des degrés) soit incalculables de façon distribuée (centralité d'intermédiarité).

Dans ce rapport, nous introduisons une nouvelle forme de centralité : la centralité du second ordre. Celle-ci peut être calculée de manière totalement distribuée et utilise une marche aléatoire parcourant le réseau de manière débiaisée. Elle procure à chaque noeud un indicateur de son importance dans le réseau.

Pour cela, chaque noeud conserve les temps écoulés entre deux visites de la marche (temps de retour) et calcule l'écart type de ces temps. À l'aide de simulations et d'une analyse théorique, nous montrons que cet écart type peut être utilisé pour identifier précisément les noeuds critiques. Il permet en outre de caractériser globalement la topologie d'un graphe donné, de manière totalement distribuée.

Mots-clés : Centralité, Marches aléatoires, Centralité d'intermédiarité, Marches aléatoires débiaisées discrètes, Temps de retours.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Second order centrality : distributed assessment of nodes criticity in complex networks

Abstract:

A complex network can be modeled as a graph representing the "who knows who" relationship. In the context of graph theory for social networks, the notion of centrality is used to assess the relative importance of nodes in a given network topology. For example, in a network composed of large dense clusters connected through only a few links, the nodes involved in those links are particularly critical as far as the network survivability is concerned. This may also impact any application running on top of it. Such information can be exploited for various topological maintenance issues to prevent congestion and disruptance. This can also be used offline to identify the most important nodes in large social interaction graphs. Several forms of centrality have been proposed so far. Yet, they suffer from imperfections : designed for abstract graphs, they are either of limited use (degree centrality), either uncomputable in a distributed setting (random walk betweenness centrality). In this paper we introduce a novel form of centrality : the second order centrality which can be computed in a fully decentralized manner. This provides locally each node with its relative criticity and relies on a random walk visiting the network in an unbiased fashion.

To this end, each node records the time elapsed between visits of that random walk (called return time in the sequel) and computes the standard deviation (or second order moment) of such return times. Both through theoretical analysis and simulation, we show that the standard deviation can be used to accurately identify critical nodes as well as to globally characterize graphs topology in a fully decentralized way.

Key-words: Centrality, Random Walk Betweenness, Unbiased Discrete Time Random Walks, Return times.

1 Introduction

Large scale networks, as organizational/social contacts, peer-to-peer, grids or wireless sensors networks often exhibit complex and huge interaction graph structure. The scale of these graph is such that it usually prevents to compute any global characteristic aggregated from individual nodes [3, 20]. Consequently, designing fully distributed solutions (in which network components participate only based on local or close neighborhood information) is of the utmost importance.

The offline analysis of complex social networks has been addressed by physicians and sociologists for many years. These works provide algorithms and metrics to extract characteristics on the interactions captured and the importance of individuals in these graphs. The notion of *centrality* [6, 7, 8, 14, 16, 25] typically provide such infomations. Existing algorithms computing node centrality often exhibit a high complexity, and can hardly be decentralized at a reasonable cost. This is a major issue as distributed systems can precisely greatly be empowered with such interaction analysis. In addition, it is becoming more and more difficult for single computers to accommodate the growing size of distributed systems. Also, new forms of decentralized systems, such as Dark Nets or crypted peer-to-peer networks simply forbid the access to the full graph topology. Finally, the trend in industry is to exploit at best the free computing power of clients instead of investing into expensive server farms. There is a stringent need to come up with fully decentralized approach adapted to such scales.

To overcome these issues, we introduce a novel notion of centrality, called *second order centrality*. The second order centrality captures the importance of individual nodes in a given topology and it allows global characterisation of a complex network. In addition this algorithm can be easily implemented in a fully decentralized way, thus accommodating the growing scale of complex networks. In this paper we make the following contributions ; *(i)* we define a novel notion of centrality, called the second moment centrality and show that this represent a meaningful metric to characterize both the criticity of individual nodes in a given topology as well as the global *health* of a complex network with respect to connections. Typically the second order centrality enables to capture the importance of nodes in a graph topology and can be used to identify critical nodes and clusters. We show this both analytically and through simulations. *(ii)* We provide a lightweight algorithm to compute in a fully decentralized way the second order centrality of each node. The strength of this algorithm lies in its simplicity. It relies on a random walk visiting the network. Nodes compute their second order centrality by simply recording the return times of that walk. The standard deviation of those return times is at the core of our approach. *(iii)* We show that this algorithm can be used to provide graph signatures and therefore information about the global characteristics of a graph. *(iv)* We provide some simulation results that not only accurately match the analysis but also provides some evidence of the relevance of that metric in a practical setting.

The rest of this paper is organized as follow. Section 2 provides the design rationale of the second order centrality. Section 3 review former forms of centrality and emphasizes their characteristics and limitations. Section 4 describes the second order centrality algorithm. The analysis is provided in Section 5. Simulation results are provided in Section 6. Finally, in Section 7, we discuss the convergence time of our approach before concluding.

2 Design rationale

We consider an arbitrary network, represented as an undirected graph $\mathcal{G} = (V, E)$, with n vertices and m edges. For a node $i \in V$, Γ_i denotes its set of neighbors in $\mathcal{G}$ (vertices with an adjacent edge to i), and d_i its degree, namely the size of Γ_i. Note that the resulting graph may represent any peer-to-peer, grid, social, or physical network. We assume a connected graph ; this is crucial to avoid a result on the connected component where the algorithm is started.

We use a *random walk* on the graph $\mathcal{G}$, *i.e.* a process progressing in the network from a node i to another node chosen uniformly in i's neighborhood Γ_i. We consider this walk to be *permanent* for it has no stop condition ; this is one of the main differences with most of random walk based algorithms (e.g. sampling [15] or search [22]). We assume that the random walk is initiated by an arbitrary node of the network. We also assume, for the sake of comprehension, that the network is static *i.e.* the topology does not change during the process execution. Finally, the random walk is assumed to be never lost.

Our approach relies on the fact that the relative importance of a given node can be inferred from the regularity at which the random walk visits the node. Typically, the algorithm exploits the time elapsed between two consecutive visits of such random walk on a given node (called return time hereafter). These return times can be either absolute time, thus implying a clock on every system node, or simply the number of steps proceeded by the random walk (which thus carries that information). Note that in the first case, nodes' clocks do not need to be synchronized, as we are only interested in local standard deviation of the return times.

Let us notice that our model only requires one random walk for the whole system to provide the algorithm result, as opposed *e.g.* to one random walk launched by each node for its own purposes.

3 Related work

In this section, we first review some notions providing global graph characteristics. Second, we consider various notions of centrality, used to assess the relative importance/criticity of nodes in a given graph. Finally, we detail the random based betweenness centrality for it is the closest approach to the second order centrality defined in this paper.

3.1 Macro level : Graph connectivity characteristics

The global connectivity characteristics of a graph can be expressed by the notions of *spectral gap* (noted λ_2), and *conductance* (noted Φ). The value λ_2 is the smallest positive eigenvalue of the Laplacian matrix of $\mathcal{G}$, L ($L = D - A$, with D and A respectively the diagonal matrix of degrees and the adjacency matrix of $\mathcal{G}$, with the sort $0 < \lambda_2 \leq ... \leq \lambda_n$). The conductance Φ is defined as follow :

$$\Phi := \inf_{C:|C|\leq N/2} \frac{E(C, \overline{C})}{|C|},$$

where $E(C, \overline{C})$ denotes the number of edges in the graph $\mathcal{G}$ between the set of nodes C, and the complementary set $\overline{C}$ (see e.g.[23]). Finally, λ_2 and Φ are

linked by the Cheeger inequality [23] that states that $\lambda_2 \geq \frac{\Phi^2}{2\Delta(\mathcal{G})}$, where $\Delta(\mathcal{G})$ is the maximal degree of nodes in the graph. Therefore the lowest the values λ_2 and Φ, the higher the probability that algorithms that are run on top of the network behaves poorly, and/or slowly. An example of particular interest is the *mixing time* of random walks, namely the number of steps that requires a random walk so that it has a uniform probability to be on any node of the given graph ; the mixing time is typically controlled by the graph conductance (see *e.g.* [21]). In short, the values of spectral gap and conductance simply express connectivity issues or bottlenecks at the granularity of the whole graph without identifying specifically involved nodes.

3.2 Micro level : individuals in the graph

Beyond global characteristics, we are interested in the properties of individual nodes and their impact on overall graph connectivity. This impact is reflected by *centrality* indices.

The *betweenness centrality* [14, 6, 7, 16] is considered the most relevant in that context. It consists in computing on each node the fraction of shortest paths that passes through it. Formally, the betweenness centrality for a node v is $b_C(v) = \Sigma_{s,t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$, where $\sigma_{s,t}(v)$ is the number of shortest paths from node s to node t passing through v, and $\sigma_{s,t}$, the total number of shortest paths from s to t. The original algorithm requires $\Omega(n^3)$ time steps to complete ; another approach completes in $O(nm)$ steps [7]. Finally, recent experimental studies [8, 16] propose some approximations for practical use in large networks.

3.3 Random walk based betweenness centrality

Despite a great interest towards the previous metric, Newman showed [25] that the notion of betweenness centrality suffers from some imperfections as it considers only nodes involved in shortest paths. A typical example is presented on Figure 1, where two clusters are linked by a few nodes only. Here, node c, which is outside the shortest paths linking left and right clusters is given a very low score of betweenness centrality despite its clear importance for alternative paths.

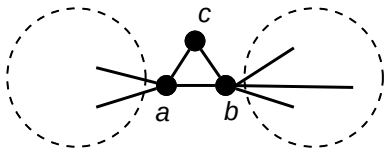


FIG. 1 – An example where betweenness centrality give node c a low score despite its importance

Such left out nodes can also be of a vital importance for the network resilience, for load balancing or facing failures of shortest path nodes. Another metric, known as *flow betweenness*, also suffers from a similar drawback [25]. *Random walk betweenness* has been introduced to fix this issue : the idea is

that a random process also takes into account non-optimal paths. Assuming the knowledge of the whole graph, the proposed method uses its adjacency matrix and completes in $O((m+n)n^2)$ steps. It consists in launching a random walk from each node s to every other node t. The random walk betweenness of a node i is equal to the number of times that a random walk starting at s and ending at t passes through i along the way, averaged by all possible (s,t) pairs. A heuristic is also added to avoid counting back and forth random walk passages on nodes.

Despite a proper definition of the random walk based centrality, the Newman's approach requires a global knowledge of the graph and this prevents its application in a fully decentralized setting.

Instead, we define a new form of centrality, called the *second order*, as opposed to Newman's approach interested in the first order (expected number of visits to a given node for all source-target pairs [25]). This centrality is computed through the use of a single random walk, in a fully decentralized manner. Each node is required to be aware of its direct neighborhood Γ only. There is no need for any global information. Instead, each node only records the random walk return times and computes the standard deviation of the stored values.

4 Second order centrality

In this section, we first describe the intuition behind our approach. We then present the distributed algorithm which outputs on each node its centrality, the value of which reflects the relative importance of that node in the network.

4.1 The high clustering intuition

To illustrate our purpose, we voluntary consider an extreme setting, known as the *barbell graph*. The following barbell graph is composed of two fully connected components (called bells) of m_1 nodes each, connected by a path of m_2 nodes. Figure 2 depicts such a graph with $m_1 = 5$ and $m_2 = 2$.

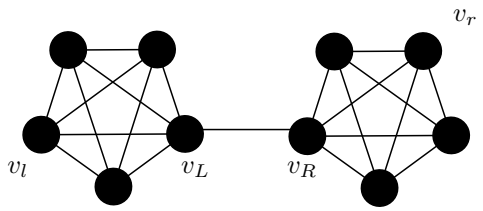


FIG. 2 – Example of a Barbell graph

Consider a random walk visiting the network from node to node. First, let us consider node v_L on Figure 2 : if a random walk is running in the left bell, v_L has the same probability $1/m_1$, than any other node in this bell to be visited by the random walk at each step. Yet, once the random walk has passed to the right bell, v_L is the mandatory passage point for the walk to get back to the left bell. Therefore, such bridge nodes, v_L and v_R, are visited more *regularly* than other nodes by a random walk continuously running. This is turned into a reduced standard deviation of the number of steps needed for a random walk to

return to them, after an initial passage (called *return time*). Our claim is then that different roles of nodes in the topology can be inferred from the return times.

Secondly, return times can also be used to discover topology issues. It can easily be shown (see e.g. [4]) that a random walk starting at any node in the left bell, say v_l, takes as mean time $m_1^2 m_2$ steps to reach another node, v_r, in the right bell (and conversely). On the other hand, nodes from a given bell are visited by the random walk often as long as the random walk remains in the same bell. Once it has crossed the path to the other bell, the trends reverses. By simply comparing the standard deviation of return times of the random walk, every graph node can then locally detect the presence of critical paths, or traps, in the topology.

4.2 Distributed second order algorithm

Based on those observations, we propose a new centrality, the second order centrality, along with a lightweight distributed algorithm to compute it, in which each node simply computes the standard deviation of the return times of a permanent unbiased random walk running on the topology.

4.2.1 Unbiased random walk

The paper by Newman considered *simple* random walks over the graph's transition probability matrix. This represents the classic process where at each node, a random walk is directed to a neighbor of the node, picked uniformly at random. Such a simple forwarding process obviously favors high degree nodes, as its *stationary distribution* $\pi_i = d_i/2m$ (see e.g. [24]). The more a node is connected the more often it is visited. Newman points out this issue and shows experimentally that it results in a correlation between a node's degree and its random walk based betweenness centrality, even for nodes that are not central in the topology.

To overcome this issue and cope with heterogeneous distribution of degree, the proposed distributed second order algorithm, relies on an *unbiased* random walk, where π_i is $1/n$ for all $i \in V$. Informally, this means that after a sufficient number of steps (called *mixing time* in literature), the random walk has an equal probability to be on any graph node.

Unbiasing the random walk ensures that the only cause of the variations of return times on nodes is due to their relative importance in the topology, and does not depend on local factors such as node degrees. In addition, the fact that the random walk eventually visits all nodes an equal number of times, speeds up the algorithm's convergence by evenly providing return times to all nodes. We used the Metropolis-Hastings technique ([17, 26]) to unbias the random walk. The node hosting the random walk selects a neighbor uniformly at random : the random walk is forwarded to the chosen neighbor with a probability depending on the degree of both nodes. This process is described from line 1 to line 9 in the algorithm description (Algorithm 1).

Algorithm 1 Second order centrality algorithm

Upon reception of the random walk on node i:
  / Metropolis-Hastings random walk */*
  Choose a neighbor j from Γ_i uniformly at random
  Query j for d_j
  Generate a random number $p \in [0, 1]$ uniformly
  if $p \leq d_i/d_j$ **then**
    forward the random walk to j
  else
    random walk remains at i
  / Standard deviation */*
  if first visit of the random walk on i **then**
    Create array Ξ_i
  else
    Compute return time r since last visit
    Add r to Ξ_i
    if $|\Xi_i| \geq 3$ **then**
      Compute standard deviation :
    $\sigma_i(N) = \sqrt{\frac{1}{N}\Sigma_{k=1}^{N}\Xi_i(k)^2 - [\frac{1}{N}\Sigma_{k=1}^{N}\Xi_i(k)]^2}$

4.2.2 Standard deviation of return times

The key point of the algorithm is the variation of the frequency at which a random walk visits nodes. Every node i in $\mathcal{G}$ joins the process on the first visit of the random walk, by creating an array Ξ_i that logs every return time. Recall that the return time to node i is defined by the time, for the walk starting at i, to return to i. We denote by $\Xi_i(k)$ the k-th return time of the random walk to node i. A simple solution to capture an irregularity of visits on nodes is to proceed as follows : after the third recorded return time, a node i computes the standard deviation

$$\sigma_i(N) = \sqrt{\frac{1}{N}\Sigma_{k=1}^{N}\Xi_i(k)^2 - [\frac{1}{N}\Sigma_{k=1}^{N}\Xi_i(k)]^2}$$

of the N values in Ξ_i. These return times being independent, we have from the strong law of large numbers :

$$\lim_{N \longrightarrow \infty} \sigma_i(N) = \sigma_i$$

Once the random walk has run for a sufficiently long time, σ values represent the relative importance of nodes in the graph : the lower the value, the higher the impact of a node. The description of the algorithm is provided in Algorithm 1.

4.2.3 Algorithm convergence time

Each node needs to be visited a few times by the random walk to compute meaningful deviation results. Therefore the algorithm convergence time is related to the *cover time* of graph $\mathcal{G}$. Cover time is defined as the number of steps needed by a random walk to visit each vertex of $\mathcal{G}$. Feige [11] showed that cover time, for a simple random walk, ranges from $(1 + o(1))n \ln n$ steps for a complete undirected graph to at most at $\frac{4}{27}n^3 + o(n^3)$ [12] for the *lollipop* graph

(a fully connected graph of $\frac{n}{2}$ nodes, linked to a line of the remaining nodes). This lower bound result holds for an unbiased random walk as all nodes in a complete graph have the same degree. We are not aware of an upper bound for an unbiased random walk. Therefore, our algorithm requires the number of steps to cover the graph, times a constant, so that each node is visited several times.

Other random walk based algorithms (*e.g.* for graph connectivity assessment [13]) exhibit running time of $O(n^3)$ in worst case of input graph irregularity ; we expect far more reduced running times for realistic networks.

5 Return times in Markov chains

This section provides some theoretical analysis of the distributed second order algorithm. More specifically we provide a formula to compute the theoretical standard deviation of return times for any node, given an input graph as a transition probability matrix. In addition, it is used as the baseline centralized theoretical prediction against which the simulation results of the distributed algorithm are compared.

This formula is also useful for *(i)* a system administrator who wants to predict the behavior of random walks over a particular graph structure before deploying it, or for *(ii)* graph nodes to derive an expected algorithm completion time when basic properties of the graph are known.

5.1 Standard deviation of return times

We look for a formula that provides standard deviation of return times on a particular node, given by the graph transition probability matrix. This return time is function of the position of this node in the graph. This idea is at the core of our approach, and is generally forgotten due to the fact that literature often provides bounds to return times that hide the local variations nodes may experience (big O notation). Those potentially small variations suffice to differentiate nodes with respect to their position in the graph.

We use a classical discrete time Markov chain model to represent the random walk running on the input graph. States of the Markov chain are nodes, or vertices V of $\mathcal{G}$. The general case of a biased walk is presented first, as an unbiased walk is simply a subcase of it. Finally, for the purpose of the demonstration and to give theoretical results on return times, we consider the transition probability matrix of the considered graph ; as precised in Section 2. This global knowledge of the graph is obviously not assumed in our distributed algorithm proposal. Proofs are deferred in the appendix.

Let $X = \{X_n,\ n \in \mathbb{N}\}$ be a homogeneous and irreducible discrete time Markov chain on the finite state space S. We denote by $P = (P(i,j))_{i,j \in S}$ its transition probability matrix and we are interested in the computation of the return times for every state of S. For every state $j \in S$, we denote by $\tau(j)$ the number of transitions (random walk steps) needed to reach state j, i.e.

$$\tau(j) = \inf\{n \geq 1 \mid X_n = j\}.$$

The state space S being finite and X being irreducible, X is recurrent which means that $\tau(j)$ is finite a.s. We denote by $f_j^{(n)}(i)$ the distribution of $\tau(j)$ when

the initial state of X is i, that is, for every $n \geq 1$,

$$f_j^{(n)}(i) = \mathbb{P}\{\tau(j) = n \mid X_0 = i\}.$$

$f_i^{(n)}(i)$ represents the probability, starting from state i, that the first return to state i occurs at instant n and, for $i \neq j$, $f_j^{(n)}(i)$ represents the probability, starting from state i, that the first visit to state j occurs at instant n. These probabilities are given by the following theorem [9]. For the sake of completeness, the proof of this theorem is also provided in the appendix.

Theorem 1 *For every $i, j \in S$ and $n \geq 1$, we have*

$$f_j^{(n)}(i) = \begin{cases} P(i,j) & \text{if} \quad n = 1 \\ \displaystyle\sum_{\ell \in S - \{j\}} P(i,\ell) f_j^{(n-1)}(\ell) & \text{if} \quad n \geq 2. \end{cases} \tag{1}$$

For every $j \in S$ and $n \geq 1$, we denote by $f_j^{(n)}$ the column vector containing the values $f_j^{(n)}(i)$ for every $i \in S$. For every $j \in S$, we introduce the matrix Q_j obtained from matrix P by replacing the jth column by zeros, that is

$$Q_j(i,\ell) = \begin{cases} P(i,\ell) & \text{if} \quad \ell \neq j \\ 0 & \text{if} \quad \ell = j. \end{cases}$$

We also introduce the column vector P_j containing the jth column of matrix P, i.e. $P_j(i) = P(i,j)$. Equation (1) can then be written in matrix notation as

$$f_j^{(n)} = \begin{cases} P_j & \text{if} \quad n = 1 \\ Q_j f_j^{(n-1)} & \text{if} \quad n \geq 2, \end{cases} \tag{2}$$

which leads to an easy computation of the vectors $f_j^{(n)}$. We now define the matrix $M = (M(i,j))_{i,j \in S}$ by $M(i,j) = \mathbb{E}\{\tau(j) \mid X_0 = i\}$. $M(i,i)$ represents the expected time between two successive visits of X to state i, and, for $i \neq j$, $M(i,j)$ represents the expected time, starting from state i, to reach state j for the first time. The Markov chain X being irreducible, we have $M(i,j) < \infty$ for every $i, j \in S$ and

$$M(i,i) = \frac{1}{\pi_i},$$

where π_i is the ith entry of the probability distribution π, which is the unique solution to the system $\pi = \pi P$.

To compute all the entries of matrix M, we introduce the column vector M_j containing the jth column of matrix M, i.e. $M_j(i) = M(i,j)$ and the column vector of ones denoted by $\mathbb{1}$. These expected values are given by the following result.

Corollary 2 *For every $j \in S$, we have*

$$M_j = (I - Q_j)^{-1} \mathbb{1}. \tag{3}$$

In practice, the column vector M_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)M_j = \mathbb{1}$.

Let us consider now the second moment of $\tau(j)$. We define the matrix $H = (H(i,j))_{i,j\in S}$ by $H(i,j) = \mathbb{E}\{\tau(j)^2 \mid X_0 = i\}$. $H(i,i)$ represents the second moment of the time between two successive visits of X to state i, and, for $i \neq j$, $H(i,j)$ represents the second moment of the time, starting from state i, to reach state j for the first time. We introduce the column vector H_j containing the jth column of matrix H, i.e. $H_j(i) = H(i,j)$. These values are given by the following result.

Corollary 3 *For every $j \in S$, we have*

$$H_j = (I - Q_j)^{-1}(I + Q_j)M_j.$$

In practice, the column vector H_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)H_j = (I + Q_j)M_j$.

The standard deviation $\sigma(i)$ of the return time to state i on a target graph is thus given by

$$\sigma(i) = \sqrt{H(i,i) - [M(i,i)]^2}. \tag{4}$$

5.1.1 Unbiaised random walks

When the random walk is unbiased, all the nodes in the graph have the same degree d (Metropolis-Hastings method virtually adds self-loops to poorly connected nodes to adjust their degree) and $P(i,j) = 1/d$ if nodes i and j are connected in the graph and 0 otherwise. This means in particular that matrix P is symmetric and thus bistochastic, i.e. $\mathbb{1}^t P = \mathbb{1}^t$, where t denotes the transpose operator. We then have $\pi_i = 1/|S|$ and $M(i,i) = |S|$. For what concerns the second order moments $H(i,i)$ of return times to state i, we have from Corollary 3,

$$(I - Q_j)H_j = (I + Q_j)M_j.$$

Premultiplying by 1^t, we get

$$\mathbb{1}^t H_j - \mathbb{1}^t Q_j H_j = \mathbb{1}^t M_j + \mathbb{1}^t Q_j M_j. \tag{5}$$

By definition of Q_j, we have $\mathbb{1}^t Q_j = \mathbb{1}^t - e_j$, where e_j is the jth unit row vector, i.e. $e_j(i) = 1$ if $i = j$ and 0 otherwise. So equation (5) simplifies as

$$\mathbb{1}^t H_j - (\mathbb{1}^t - e_j)H_j = \mathbb{1}^t M_j + (\mathbb{1}^t - e_j)M_j$$

and thus

$$H(j,j) = 2\sum_{i\in S} M(i,j) - M(j,j) = 2\sum_{i\in S} M(i,j) - |S|.$$

The standard deviation $\sigma(j)$ then writes, from relation (4), as

$$\sigma(j) = \sqrt{2\sum_{i\in S} M(i,j) - |S|(|S|+1)}. \tag{6}$$

5.2 Results for 3 classes of regular graphs

We instantiate this formula for three extreme graph diameter settings : a complete graph (diameter 1), a ring (diameter $\lfloor \frac{n}{2} \rfloor$) and a line (diameter n). On all graphs, we consider an unbiaised random walk.

On the ring we have $d = 2$ and the non zero transition probabilities are given, f or every $i \in S = \{0, \ldots, n-1\}$, by

$$P(i, i+1 \ (mod\ n)) = P(i, i-1 \ (mod\ n)) = 1/2.$$

The standard deviations of the return times are given by the following theorem.

Theorem 4 *For the unbiased random walk on a n nodes ring, we have $\sigma(j) = \sigma$ for every j where*

$$\sigma = \sqrt{\frac{n(n-1)(n-2)}{3}}.$$

Proof. It is easily checked that the solution to equation (3) is given, for $i \neq j$, by

$$M(i, j) = (n - |i - j|)(|i - j|)$$

and as mentioned above, we have $M(i, i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i, j) = n + \frac{(n-1)n(n+1)}{6}$$

Using equation (6), we obtain the desired result. ■

Note that the fact all the $\sigma(j)$'s are equal is due to the regularity of the structure.

On the complete graph, we have $d = n - 1$ and thus, the transition probabilities are given, for every $i, j \in S$, by $P(i, j) = 1/(n-1)$ if $i \neq j$ and $P(i, i) = 0$. The standard deviations of the return times are given by the following theorem.

Theorem 5 *For an unbiased random walk on a n nodes complete graph, we have $\sigma(j) = \sigma$ for every j where*

$$\sigma = \sqrt{(n-1)(n-2)}.$$

Proof. It is easily checked that the solution to equation (3) is given, for $i \neq j$, by

$$M(i, j) = n - 1$$

and as mentioned above, we have $M(i, i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i, j) = n + (n-1)^2$$

Using equation (6), we obtain the desired result. ■

Again the fact all the $\sigma(j)$'s are equal is due to the regularity of the structure.

On a line we have $d = 2$ and the non zero transition probabilities are given, for every $i \in \{1, \ldots, n-2\}$, by

$$P(i, i+1) = P(i, i-1) = 1/2$$

and $P(0,0) = P(0,1) = P(n-1, n-2) = P(n-1, n-1) = 1/2$. The standard deviations of the return times are given by the following theorem.

Theorem 6 *For the unbiased random walk on a n nodes line, we have for every* $j \in \{0, 1, \ldots, n-1\}$,

$$\sigma(j) = \sqrt{\frac{n(n-1)(4n-5)}{3} - 4nj(n-j-1)}.$$

Proof. It is easily checked that the solution to equation (3) is given, for $i \neq j$, by

$$\begin{array}{rcl} M(i,j) & = & (j-i)(i+j+1) \text{ for } i < j \\ M(i,j) & = & (i-j)(2n-(i+j+1)) \text{ for } i > j \end{array}$$

and as mentioned above, we have $M(i,i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i,j) = \frac{n(2n^2-3n+4)}{3} - 2nj(n-j-1).$$

Using equation (6), we obtain the desired result. ■

Note that the fact that $\sigma(j) = \sigma(n-j-1)$ is due to the symmetry of the structure, and that $\sigma(j)$ is minimal for nodes $\lfloor \frac{n-1}{2} \rfloor$ and $\lceil \frac{n-1}{2} \rceil$. This last remark highlights the fact that our algorithm produces a centrality result, as on a line, critical nodes (*wrt* centrality) are in the middle of it.

Moreover, in the context of symmetric graphs, the ring and the complete graph are extreme cases of resiliency : the ring is the weakest (it is only 2-connected), whereas the complete graph is the most robust structure (it is $(n-1)$-connected). We believe it is reasonable to expect the standard deviation of symmetric graphs to evolve between the corresponding values (*i.e.* between $o(n)$ for the complete graph, and $o(n^{3/2})$ for the ring).

5.3 Application to specific graphs

Another contribution of this paper is, through the previous formula (4), to be able to provide *signatures*[1] of graphs. We now expose different signatures, that can help to sort graphs according to their *health* : a good health is characterized by good navigability properties. In this section, we illustrate this by computing this result for several well known graph topologies.

The σ value is computed for each node, based on the transition probability matrix (using a random walk unbiased with the Metropolis-Hastings method).

We considered the following graphs for theoretical computation : *(i)* a *random graph*, constructed on the Erdös-Rényi model [10]. The probability that two

[1] a graph signature could be seen as a footprint constituted by the distribution of standard deviation values of its nodes

edges are connected is given by $p = \frac{\theta \ln n}{n}$. Probability $p > \frac{\ln n}{n}$ insures connectivity of $\mathcal{G}$, *i.e.* that no vertex is isolated. *(ii)* a *clusterized graph* composed of two equal size random graphs linked by a single edge. The resulting graph is close to a Barbell graph presented earlier in the paper, with the difference that left and right bells are not fully connected (*i.e.* not complete graphs). *(iii)* a *ring lattice*, where a node i is connected to nodes $i - k/2, i - k/2 + 1, \ldots, i + k/2$ (values mod n, and k being an input parameter), excluding itself. Finally *(iv)* a *scale-free* graph, based on the Barabási/Albert model [3]. We believe that those graphs are representative of classical graph families, *i.e.* graphs that are both widely studied and often targeted in today network designs, or parts of actual social graphs [28]. All four graphs have a size of 10^3 nodes, and all input parameters have been set so that their average degree is 20, insuring a fair comparison.

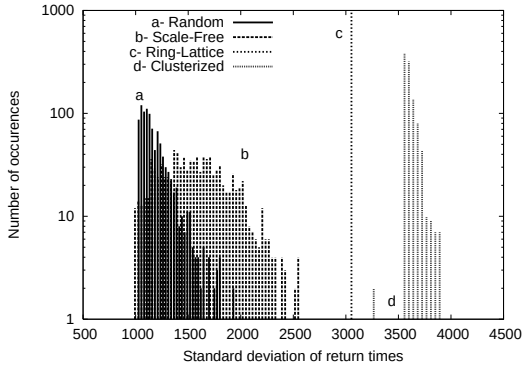


FIG. 3 – Histogram of theoretical standard deviation of return times, for 4 particular graphs

Results are presented as histograms on Figure 3. A particular point, say $(x = 2500, y = 3)$, expresses that three network nodes have a resulting σ of 2500. We first observe a clear difference in the distribution of results for the tested graphs. A thin distribution of values for a particular graph basically means that all nodes have a similar role or importance in the structure. Contrarily, a significant scattering in the values is to be interpreted as an important irregularity of roles. Furthermore, in graphs of equal sizes and degree average, differences in tendencies of mean σ value, reflect discrepancies in the navigation properties of those graphs (related to diameter or presence of bottlenecks).

The graph exhibiting the lowest σ values is the random graph (marked a), as the gathered values on nodes lie approximately in a $[1000 : 1500]$ step interval. This matches the consensus about the attractive properties (low diameter and low clustering coefficient) of such graphs. It is followed by the Barabási/Albert graph (b), which has a larger repartition of values $(1000 - 2500)$. This is due to the fact that hubs (highest degree nodes) are part of a lot of shortest paths, and that most of other nodes have a far inferior global importance. In the ring lattice (c), a line-shaped distribution (note the logscale on the y-axis) is observed, due to the perfect regularity of the lattice structure. Finally, the clusterized graph (d) exhibits an interesting distribution for two reasons. First, this structure, composed of two random graphs linked by an edge, produces a σ value, three

time the one of a (single) random graph. This is obviously due to the difficulty of the random walk progression in the structure, as exposed in Section 4.1. The second observation concerns the detached line around 3250 : the two nodes responsible of this line are the bridge nodes of the structure (as nodes v_L and v_R on Figure 2). This confirms the intuition given in 4.1 stating that those bridges play an important role in the topology and thus see the random walk more regularly than other nodes.

To summarize, this formal method can also be leveraged to assign particular signatures to class of graphs, allowing to sort them by order of healthiness or usability in practice.

6 Evaluation

We now evaluate our distributed algorithm through extensive simulations on various graph topologies, including the previously introduced ones.

In this section, we report the experimental results. We evaluate the distributed second order centrality algorithm along the following metrics : *(i)* the ability to produce unbiased results in the presence of heterogeneous degree distributions ; *(ii)* the matching between the theoretical expectations and the experimental results with respect to convergence time and the ability to detect topology critical nodes, and finally *(iii)* the practicality of the approach : typically we show that for graphs used in practice, convergence time is, as expected, far less than the upper bounds given by theory.

We use the theoretical values computed and reported previous section, as baseline for comparison. We experimentally show that our algorithm results on nodes converge within a small accuracy window, thus validating our proposal. We then give some intuitions to show that the computed values can be used to trigger some distributed repair mechanisms, or to identify the existence of clusterized parts in the graph.

Experiments have been obtained using the PeerSim discrete event simulator [1].

6.1 Conductance and centrality : micro example

As we mentioned earlier, the conductance of a graph could be related to some critical bottleneck nodes, due to poorly connected parts. Here, we illustrate our algorithm on a micro graph to compute the conductance ; we also provide theoretical and simulated results of our algorithm, to compare both notions.

Figure 4 plots a typical run of the algorithm, with the standard deviation of random walk's visits on every node (italic values). The bold values are the theoretical values of standard deviation provided by formula (4). The conductance Φ has been computed for every adjacent vertices of this small example ($\Phi = 0.2$) ; we also present the 2 other smallest values and the cut implied (dashed lines). Note that the conductance of the graph is the minimum of the values of the cuts.

Nodes at the edges of the graph have a relatively high σ, and nodes in the middle of the graph the lowest value, following the intuition that the random walk's visits are less irregular for centered nodes compared to visits on edge nodes, on that example close to a *line* or *path graph* (thus matching Theorem 6).

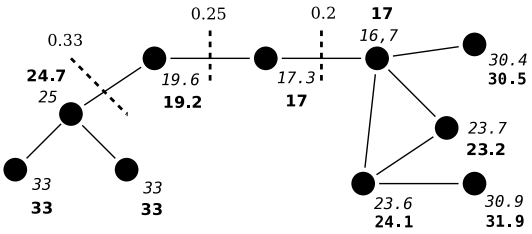


FIG. 4 – Conductance and standard deviation of return times on a micro network with n=10 nodes

We observe that the importance of nodes in the topology is effectively correlated to the inverse of their value order : nodes with the smallest σ values are attached to the conductance cut with the smallest value.

6.2 Degree bias removal

We now provide a simple algorithm run example, aiming to assess the effectiveness of the Metropolis-Hastings method used jointly to our algorithm, on our typical clusterized graph. Figure 5 depicts simulation results of the proposed algorithm, over the previously introduced clusterized graph ($n = 10^3$, $p = \frac{1.5 \ln n}{n}$). We run a simple random walk instance, and the unbiased version ; results are plotted after 2.10^6 random walk steps.

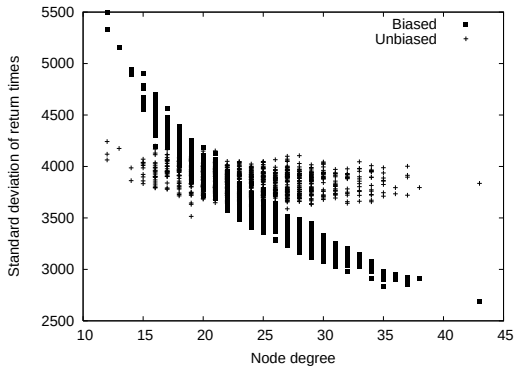


FIG. 5 – Simple VS unbiased random walk centrality estimation

We observe that for the simple random walk case, there is a clear correlation between resulting standard deviations and nodes' degree. Recall that low σ significates a high importance in the topology ; high degree nodes then all gets a high value, despite a non necessarily real importance. Contrariwise, no effect is measured on the unbiased case, as the σ are concentrated in a tighter range, that does not decrease when nodes' degree is increasing. This example confirms that our algorithm removes the fact of considering a node with a high degree as more important than it really is.

Note that using an unbiased random walk preserves the importance of high degree nodes, if this high degree is correlated to its importance (for example a node may be some kind of hub and is therefore lying on many shortest paths [25]).

6.3 Speed of convergence

This section studies how fast simulations match the theoretical expectations provided using formula (4).

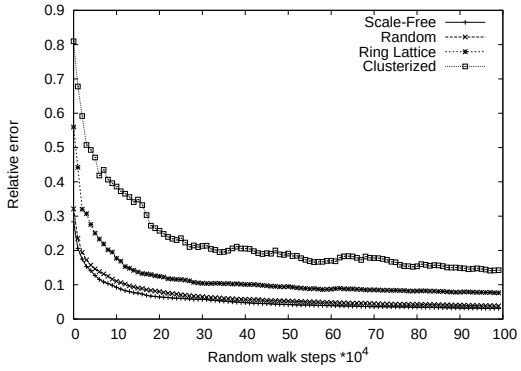


FIG. 6 – Convergence of algorithm runs towards theoretical prediction, for 4 particular graphs

Figure 6 plots, for the four previously introduced graphs, the error ratio of algorithm results as the random walk proceeds. The y-axis represents the computation error, that is the algorithm result value minus the theoretical one, normalized by the theoretical value ($y = 0$ then exhibit a perfect result). Presented curves are the average result of all graph nodes' value.

As the random walk progresses in the graph, the number of visits on nodes strictly increases, and thus give a larger set of return time values in their Ξ array. The computation of standard deviations then provides continuously improved estimations of importance[2]. We note the algorithm quickly converges to a small error window, validating its behavior against theory prediction and showing a relatively fast behavior (compared to the worst case $O(n^3)$ theoretical prediction). This reflects the theoretical analysis provided in Section 5.3 : the largest the diameter and the more cluserized the graphs, the longer the convergence of our algorithm toward an acceptable value. In high diameter/clusterized settings, the random walk process may get "trapped" in specific zones. Escaping from such zones may take time, thus slowing down the computation of an acceptable standard deviation.

6.4 Bottleneck nodes

We now focus on a specific goal of the proposed algorithm, namely detecting bottleneck. To this end, we consider the most critical nodes of the cluserized

[2]Note that we assume here unlimited memory as we believe this is not an issue in practice for the storage of simple integers

topology : the bridges. Those nodes are likely to receive a high pressure from applications that are run on top of the topology, due to their critical positions.

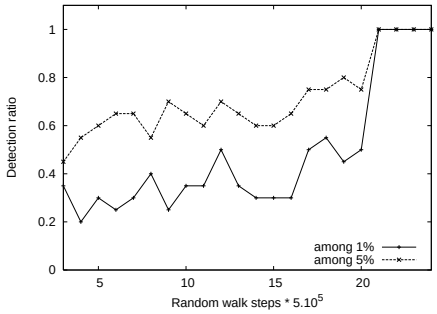


FIG. 7 – Evolution of the position of bridges among the lowest standard deviations of all network nodes

Settings of the experiment being $n = 2 \times 500$ random graphs linked by an edge, we focus on the σ values of the two nodes sharing the edge that gathers the two graphs (nodes v_L and v_R of Figure 2). We study the σ of those nodes, compared to those of all network nodes. In other words, we check out if their values are effectively among the smallest values given by our algorithm, thus assessing their effective criticity.

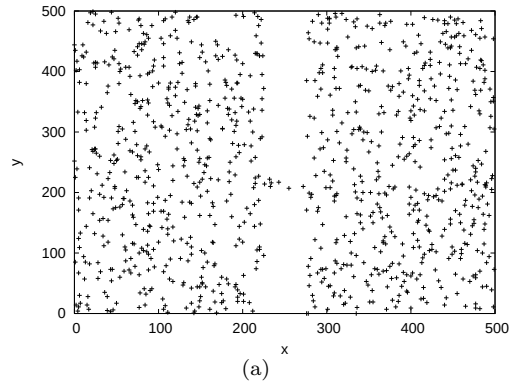
Figure 7 depicts the effective position of those bridges in either 1 or 5% of the smallest σ values of all nodes in $\mathcal{G}$ as a function of the number of random walk steps. The results shown are averaged over 20 experiments. We observe a convergence of the algorithm toward an accurate ordering, after an initial bootstrap phase of the algorithm.

Figure 8(b) provides a visual example of the distribution of those values on all system nodes. The considered network, Figure 8(a), is a 2-Dimensional network, on the model of a wireless sensor network for example, where nodes are connected to their close geographic neighbors. Results of our algorithm are plotted on Figure 8(b), where darker zones correspond to low standard deviations while clearer zones corresponds to higher values of σ. It is clear from this figure that nodes at the edges of the topology are visited irregularly, and nodes on the bridges have low values, as they are critical passage points. An animation of the evolution of the σ values on nodes is available at [2] ; snapshots of the network are taken every 25.10^3 steps.

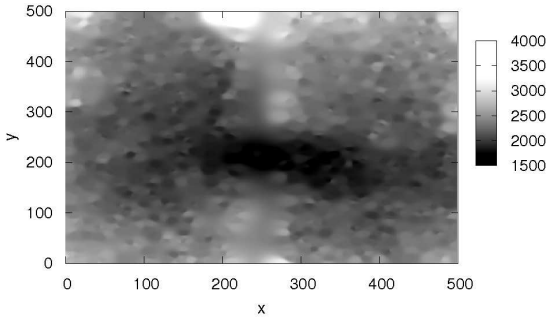
Those obvious differences in the ranking of nodes with respect to their σ, may be used to detect, in a distributed fashion, those central and potentially critical nodes by comparing such values.

6.5 Towards distributed detection

In online applications of this algorithm (as opposed to offline graph analysis), a target standard deviation σ_{ideal} may be provided to each node so that it is able to estimate its position compared to the ideal situation captured by σ_{ideal}. Computing its own standard deviation σ, a node can compute the ratio $\frac{\sigma}{\sigma_{ideal}}$, triggering repair if this ratio deviates from a predefined threshold, in



(a)



(b)

FIG. 8 – (a) Repartition of 10^3 nodes on a bottlenecked 2-D topology, (b) Resulting distribution of standard deviations on nodes, after 1×10^6 steps

a totally distributed fashion. Repair includes rewiring, that is neighborhood re-arrangement by creating new edges or dropping some others, depending on node's criticity. σ_{ideal} can be computed using tools provided in Section 5, or through simulations on graph models.

Nodes might also leverage each other information to improve their perception of the graph characteristics. Nodes can detect the presence of clusters simply by exchanging their return time array. Suppose that nodes a and b exchange their sets of return times Ξ_a and Ξ_b. The ratio $r_{a\to b} = \frac{\sigma(\Xi_a \cup \Xi_b)}{\sigma(\Xi_a)}$ can be exploited to achieve distributed cluster detection : if a and b are located in two different clusters, then the standard deviation of the union of passage times is small, so that $r_{a\to b}$ is low. Conversely, if nodes a and b are in the same cluster, the random walk is likely to hit both at very close periods, so that $r_{a\to b}$ converges to 1. As an illustration, consider a graph made of 2 clusters A and B. A random walk that starts in cluster A, remains in that cluster for 2000 hops, then gets trapped in cluster B for another 2000 hops, then returns to A for 2000 hops, and so on. A node a belonging to cluster A sees the random walk regularly in the intervals $[0 : 2000]$ and $[4000 : 6000]$. Thus, the major contribution to $\sigma(\Xi_a)$ comes from the large delay between the last visit in the first interval and the first visit in

the last interval. When a merges it's view with the view of a node b in cluster B, which sees the random walk regularly in $[2000 : 4000]$, this delay disappears, reducing significantly $\sigma(\Xi_a \cup \Xi_b)$ w.r.t. $\sigma(\Xi_a)$: the perceived difference is high. Contrarily, if b is in cluster A too, the huge $[2000 : 4000]$ delay is left intact : the difference tends to be low.

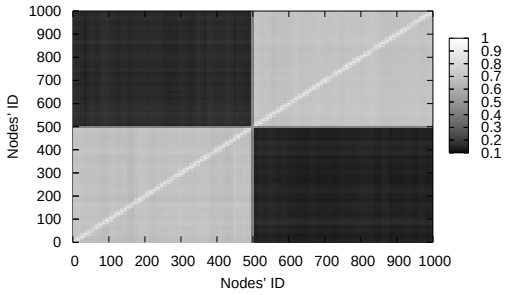


FIG. 9 – Node to node comparison of passage times, for distributed cluster detection

This is illustrated on Figure 9. In a $n = 2 \times 500$ clusterized graph, where the 500 first nodes are in a cluster and the remaining ones in the other, $r_{a \to b}$ ratio is computed with any two nodes (obviously $r_{a \to b} \equiv r_{b \to a}$, thus graphic is symmetrical). Two main colors appear (high and low difference due to merging), corresponding to the two clusters of the topology. Nodes with IDs 499 and 500 are bridges and thus have an intermediary ratio.

This illustrates that though a sampling mechanism provided by the application (*e.g.* [27, 19]), nodes would be able to detect graph degeneration into different clusters. Moreover, this allows any pair of nodes to guess whether they belong the same cluster or not. This may constitute a useful indicator to empower graph reparation algorithms.

7 The termination question

We finally discuss here a limitation of our approach as well as some other random walk based protocols. The fully distributed nature of our model implies that we do not make assumptions on the knowledge of global parameters such n, the size of the graph, or Φ, its conductance (related to random walk mixing time). As a consequence, a single node cannot decide on its own when the algorithm has converged, which means that it does not know when the random walk has run a long enough period of time to have visited few times the whole graph. Such a problem is related to the general problem of distributed termination detection. An extreme case is the Barbell graph considered on Figure 2. Assume the random walk starts in the left bell ; then node v_l has a high probability to be visited several times before the random walk passes in the right bell. If the graph was less severely degenerated, the few return times computed by v_l would have been sufficient to get a representative σ value ; in this particular case, we would like the node to take a value as a result of the algorithm when the

random walk has went several times back and forth in both bells. Convergence thus depends on network size and conductance. This problem is actually related to the mixing time of the random walk process, introduced in the related work Section. Many random walk based algorithms also suffer from this factor [15], [26].

In this light, our approach *eventually* converges to a satisfying standard deviation value, meaning a convergence after a constant factor (number of visits needed on each node) times the upper bound on cover time, that handle worst cases of topology wiring. A practical approach to this decision problem on each node is to use periodic comparison of values between nodes, as highlighted in Section 6.5.

If we relax the assumption on the non-knowledge of n, by having a rough estimation of its order of magnitude (*e.g.* for offline social graph analysis, researchers have an idea of the dataset size, or in peer-to-peer systems, designers have an idea of the popularity of their application), then worst case time to wait is directly derivable from cover time bounds.

8 Conclusion

While evaluating the global characteristic of complex network with respect to connectivity, it is also of the utmost importance to clearly identify the criticity of individual nodes. Such nodes may be at the origin of bottlenecks for example that can significantly hamper the performance of any application running on the network. In an attempt to overcome the drawback of current approaches, which are either not addressing all individual nodes or cannot scale to contemporary network sizes, we introduce a novel centrality, called the second order. Its preserves the advantages highlighted by Newman over previously introduced centralities, while being both simple, lightweight and able to be computed in a fully decentralized way. Based on our claim that regularity of visits on nodes reflects their relative importance, we showed that a single random walk, running permanently in the system, can distributedly provides values that allow a ranking in the topology. We provide theoretical analysis of the second order centrality. Simulation results match the analysis and highlights the fact that such an algorithm can be considered in practice to assess the relative importance of nodes in large scale networks.

In the light of recent work [5] on the use of multiple parallel random walks to lower cover time (k times linear speed up for large classes of graphs, for $k \leq \log n$ walks), or on the fact [18] that a small extra neighborhood knowledge can suffice to bias a random walk in order to speed up cover time ($O(n^2 \log n)$ instead of $O(n^3)$), it would be interesting as future work to study if those applications can lower convergence time of our approach, without producing significant side effects.

9 Proofs

9.1 Proof of Theorem 1

By definition of $f_j^{(n)}(i)$ we have, for $n = 1$, $f_j^{(1)}(i) = P(i, j)$. For $n \geq 2$, we have

$$
\begin{aligned}
f_j^{(n)}(i) &= \mathbb{P}\{\tau(j) = n \mid X_0 = i\} \\
&= \mathbb{P}\{X_n = j, X_k \neq j, 1 \leq k \leq n-1 \mid X_0 = i\} \\
&= \sum_{\ell \in S - \{j\}} \mathbb{P}\{X_n = j, X_k \neq j, 2 \leq k \leq n-1, X_1 = \ell \mid X_0 = i\} \\
&= \sum_{\ell \in S - \{j\}} P(i,\ell)\mathbb{P}\{X_n = j, X_k \neq j, 2 \leq k \leq n-1 \mid X_1 = \ell\} \\
&= \sum_{\ell \in S - \{j\}} P(i,\ell)\mathbb{P}\{X_{n-1} = j, X_k \neq j, 1 \leq k \leq n-2 \mid X_0 = \ell\} \\
&= \sum_{\ell \in S - \{j\}} P(i,\ell) f_j^{(n-1)}(\ell),
\end{aligned}
$$

where the last but one and the antepenultimate equalities come respectively from the Markov property and the homogeneity of the Markov chain X. ■

9.2 Proof of Corollary 2

Using Relation (2), we obtain

$$
\begin{aligned}
M_j &= \sum_{n=1}^{\infty} n f_j^{(n)} \\
&= P_j + Q_j \sum_{n=2}^{\infty} n f_j^{(n-1)} \\
&= P_j + Q_j \left(\sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} f_j^{(n)} \right) \\
&= P_j + Q_j \left(M_j + \mathbb{1} \right),
\end{aligned}
$$

and, since $P_j + Q_j \mathbb{1} = \mathbb{1}$, we get

$$M_j = Q_j M_j + \mathbb{1}.$$

Matrix Q_j is the submatrix of the transition probability matrix of an absorbing Markov chain with $|S|$ transient states and one absorbing state, thus the matrix $I - Q_j$ is invertible. This leads to

$$M_j = (I - Q_j)^{-1} \mathbb{1}.$$

■

9.3 Proof of Corollary 3

Using again Relation (2), we obtain

$$\begin{aligned} H_j &= \sum_{n=1}^{\infty} n^2 f_j^{(n)} \\ &= P_j + Q_j \sum_{n=2}^{\infty} n^2 f_j^{(n-1)} \\ &= P_j + Q_j \left(\sum_{n=1}^{\infty} n^2 f_j^{(n)} + 2 \sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} n f_j^{(n)} \right) \\ &= P_j + Q_j \left(H_j + 2M_j + \mathbb{1} \right) \\ &= Q_j H_j + 2Q_j M_j + \mathbb{1} \\ &= Q_j H_j + Q_j M_j + M_j, \end{aligned}$$

since, from Corollary 2, we have $Q_j M_j + \mathbb{1} = M_j$. This leads to

$$H_j = (I - Q_j)^{-1}(I + Q_j)M_j.$$

■

Références

[1] http ://peersim.sourceforge.net.

[2] http ://www.irisa.fr/asap/intranet/second_order.avi.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74 :47–97, 2002.

[4] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs.

[5] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. In *SPAA '08 : Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 119–128, New York, NY, USA, 2008. ACM.

[6] Marc Barthelemy. Betweenness centrality in large complex networks. *EUR.PHYS.JOUR.B*, 38 :163, 2004.

[7] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25 :163–177, 2001.

[8] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, pages 2303–2318, 2007.

[9] E. Cinlar. *Introduction to stochastic Processes*. Prentice Hall, New-Jersey, 1975.

[10] P. Erdos and A. Renyi. On random graphs. In *Publicationes Mathematicae*, pages 6 : 290–297, 1959.

[11] Uriel Feige. A tight lower bound for the cover time of random walks on graphs, random structures and algorithms 6. In *Random Structures and Algorithms*, pages 433–438, 1995.

[12] Uriel Feige. A tight upper bound on the cover time for random walks on graphs. *RSA : Random Structures & Algorithms*, 6, 1995.

[13] Uriel Feige. A fast randomized logspace algorithm for graph connectivity. *Theor. Comput. Sci.*, 169(2) :147–160, 1996.

[14] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1) :35–41, March 1977.

[15] Ayalvadi Ganesh, Anne-Marie Kermarrec, Erwan Le Merrer, and Laurent Massoulié. Peer counting and sampling in overlay networks based on random walks. *Distributed Computing*, 20(4) :267–278, 2007.

[16] Robert Geisberger, Peter Sanders, and Dominik Schultes. Better approximation of betweenness centrality. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2008.

[17] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1) :97–109, 1970.

[18] Satoshi Ikeda, Izumi Kubo, Norihiro Okumoto, and Masafumi Yamashita. Impact of local topological information on random walks on finite graphs. In *ICALP*, pages 1054–1067, 2003.

[19] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service : Experimental evaluation of unstructured gossip-based implementations. *Lecture Notes in Computer Science*, 3231 :79–98, 2004.

[20] Matthieu Latapy and Clémence Magnien. Measuring fundamental properties of real-world complex networks. *CoRR*, abs/cs/0609115, 2006.

[21] L. Lovàsz. Random walks on graphs : A survey,. In *Combinatorics, Paul Erdos is Eighty, Vol. 2 (ed. D. Miklós, V. T. Sós, T. Szonyi)*, 1998.

[22] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02 : Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.

[23] B. Mohar. Some applications of laplace eigenvalues of graphs, 1997.

[24] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[25] Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1) :39–54, January 2005.

[26] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *IMC '06 : Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 27–40, New York, NY, USA, 2006. ACM.

[27] Spyros Voulgaris, Spyros Voulgaris, Daniela Gavidia, Daniela Gavidia, Maarten Van Steen, and Maarten Van Steen. Cyclon : Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13 :2005, 2005.

[28] Jennifer Xu and Hsinchun Chen. The topology of dark networks. *Commun. ACM*, 51(10) :58–65, 2008.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399